SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 1 Data 01.03.2025 Temat: „Zaawansowane techniki analizy regresji; praktycznie zastosowanie” Wariant 2	Anna Więzik Informatyka II stopień, niestacjonarne, 2 semestr, gr.1a TTO

1. Polecenie:

   Link do repozytorium: https://github.com/AnaShiro/NoD2_2025

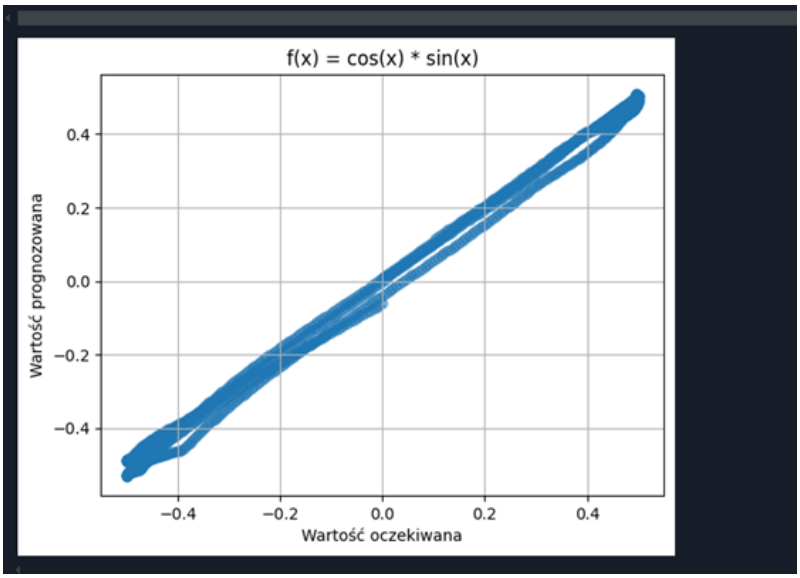
   Zadanie dotyczy modelowania funkcji matematycznych za pomocą sztucznej sieci neuronowej. Celem jest uzyskanie sieci neuronowej (poprzez zmianę zarówno liczby warstw ukrytych, jak i liczby neuronów), która spełnia warunek Error < 0.01. Wyniki modelowania należy przedstawić za pomocą wykresu punktowego, gdzie oś x reprezentuje wartość oczekiwaną, a oś y wartość przewidywaną.

   2. $f(x) = \cos(x)^{\sin(x)}, \quad x \in [0; 2\pi]$

2. Opis programu opracowanego

```python
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

X = np.linspace(0, 2 * np.pi, 1000).reshape(-1, 1)
y = np.cos(X) * np.sin(X)

model = Sequential([
    Dense(64, activation='relu', input_shape=(1,)),
    Dense(64, activation='relu'),
    Dense(1)
])

model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=300, batch_size=32, verbose=0)

y_pred = model.predict(X)

plt.scatter(y, y_pred, alpha=0.5)
plt.xlabel("Wartość oczekiwana")
plt.ylabel("Wartość prognozowana")
plt.title("f(x) = cos(x) * sin(x)")
plt.grid(True)
plt.show()
```

✓ 10.9s




```python
X = np.random.rand(1000, 4) * [200, 10, 3, 1]
y = np.random.rand(1000, 3) * [5000, 2000, 25]

model = Sequential([
    Dense(64, activation='relu', input_shape=(4,)),
    Dense(64, activation='relu'),
    Dense(3)
])

model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=100, batch_size=32, verbose=0)

loss = model.evaluate(X, y, verbose=0)
print(f"Strata końcowa (MSE): {loss:.4f}")
```

✓ 3.3s

```
Strata końcowa (MSE): 880289.3750
```

3. Wnioski

Zastosowanie sztucznych sieci neuronowych do analizy regresji okazało się skuteczne zarówno dla problemów liniowych, jak i nieliniowych. Dobrze dobrana architektura, uwzględniająca liczbę warstw ukrytych i neuronów, znacząco wpływa na zdolność modelu do odwzorowania złożonych zależności między zmiennymi. Jednocześnie nadmiar warstw może prowadzić do przeuczenia i problemów z wydajnością.

Funkcje aktywacji, takie jak ReLU, oraz techniki normalizacji, pomagają w unikaniu problemów z zanikaniem lub eksplozją gradientu podczas treningu modelu. Weryfikacja skuteczności sieci za pomocą cross-validation pozwoliła na ocenę jej zdolności generalizacyjnych. Sieci neuronowe wykazały się dużą wszechstronnością, sprawdzając się w prognozowaniu parametrów zdrowotnych, zużycia energii oraz innych wielowymiarowych zadań.

Ostatecznie, odpowiednie dobranie parametrów i architektury sieci jest kluczowe dla uzyskania niskiego błędu predykcji i wysokiej skuteczności modelu.